Software Testing and Validation – 2017/18

Instituto Superior Técnico

Vos
Project Report

Group 01 – Alameda

Francisca Cambra
ist181057

Rui Ventura
ist181045

April 11, 2018

Contents

1 Method-Scope Tests **2**

1.1 `assignPhoneNumber` . . . 2

1.1.1 Test Pattern – Category-partition . . . 2

1.1.2 Functions . . . 2

1.1.3 Input/Output Parameters . . . 2

1.1.4 Categories & Choices . . . 2

1.1.5 Constraints . . . 3

1.1.6 Test Cases . . . 3

1.2 `computeBill` method . . . 4

1.2.1 Test Pattern – Combinational Function . . . 4

1.2.2 Decision Tree . . . 4

1.2.3 Domain Matrices . . . 4

v_0 domain matrix . . . 4

v_1 domain matrix . . . 5

v_2 domain matrix . . . 5

v_3 domain matrix . . . 5

v_4 domain matrix . . . 6

v_5 domain matrix . . . 6

v_6 domain matrix . . . 6

2 Class-Scope Tests **7**

2.1 `Client` class . . . 7

2.2 `Mobile` class . . . 8

1 Method-Scope Tests

1.1 `assignPhoneNumber`

```
final void assignPhoneNumber(int clientNif, int phoneNumber) throws
↪  InvalidOperationException {
```

Assigns a free phone number to a client of *Vos* if all conditions are met. If at least one of these conditions does not hold, then this method does not change anything. In such cases, it throws an `InvalidOperationException` exception.

1.1.1 Test Pattern – Category-partition

1.1.2 Functions

- Primary function
  - Assign free phone number to a client without a number
- Secondary functions
  - Throw `InvalidOperationException` if conditions aren't met
    - Invalid nif (nif $\notin [10^8, 10^9[$)
    - Invalid phone number (number $\notin [10^8, 10^9[$)
    - Client doesn't exist (valid nif)
    - Assign a previously assigned number to a client

1.1.3 Input/Output Parameters

- Input
  - `clientNif` – The nif of the client to assign a number to
  - `phoneNumber` – The phone number to be assigned
  - `clients` – The set of *Vos* clients managed by `ClientManager`
- Output
  - `client` – The updated client, if a number was assigned successfully

1.1.4 Categories & Choices

Parameter	Category	Choices
`clientNif`	*Vos* client (w/ $\#numbers$ phone numbers)	$\#numbers \in [1, 5[$
		$\#numbers = 5$ (MAX)
	Not a *Vos* client	`clientNif` $\in [10^8, 10^9[$
	Invalid nif	`clientNif` $\notin [10^8, 10^9[$
`phoneNumber`	*Vos* phone number	Free (Unassigned)
		Not free (Assigned)
	Not a *Vos* number	`phoneNumber` $\in [10^8, 10^9[$
	Invalid number	`phoneNumber` $\notin [10^8, 10^9[$
`clients`	n-elements	$n = 0$ (Empty)
		$n \in [1, \text{MAX}]$ (Not empty)

1.1.5 Constraints

- Empty `clients` list precludes the possibility of assigning a `phoneNumber`

1.1.6 Test Cases

	Choices			Expected Result	
TC	`clientNif`	`phoneNumber`	`clients`	**Exception**	`client`
1	$\#numbers \in [1, 5[$	Free	$n \in [1, \text{MAX}]$	NO	$\#numbers \in]1, 5]$
2	$\#numbers \in [1, 5[$	Not free	$n \in [1, \text{MAX}]$	YES	—
3	$\#numbers \in [1, 5[$	$\notin [10^8, 10^9[$	$n \in [1, \text{MAX}]$	YES	—
4	$\#numbers = 5$	Free	$n \in [1, \text{MAX}]$	YES	—
5	$\#numbers = 5$	Not free	$n \in [1, \text{MAX}]$	YES	—
6	$\#numbers = 5$	$\notin [10^8, 10^9[$	$n \in [1, \text{MAX}]$	YES	—
7	`clientNif` $\in [10^8, 10^9[$	Free	$n \in [1, \text{MAX}]$	YES	—
8	`clientNif` $\in [10^8, 10^9[$	Not free	$n \in [1, \text{MAX}]$	YES	—
9	`clientNif` $\in [10^8, 10^9[$	$\notin [10^8, 10^9[$	$n \in [1, \text{MAX}]$	YES	—
10	`clientNif` $\notin [10^8, 10^9[$	Free	$n \in [1, \text{MAX}]$	YES	—
11	`clientNif` $\notin [10^8, 10^9[$	Not free	$n \in [1, \text{MAX}]$	YES	—
12	`clientNif` $\notin [10^8, 10^9[$	$\notin [10^8, 10^9[$	$n \in [1, \text{MAX}]$	YES	—

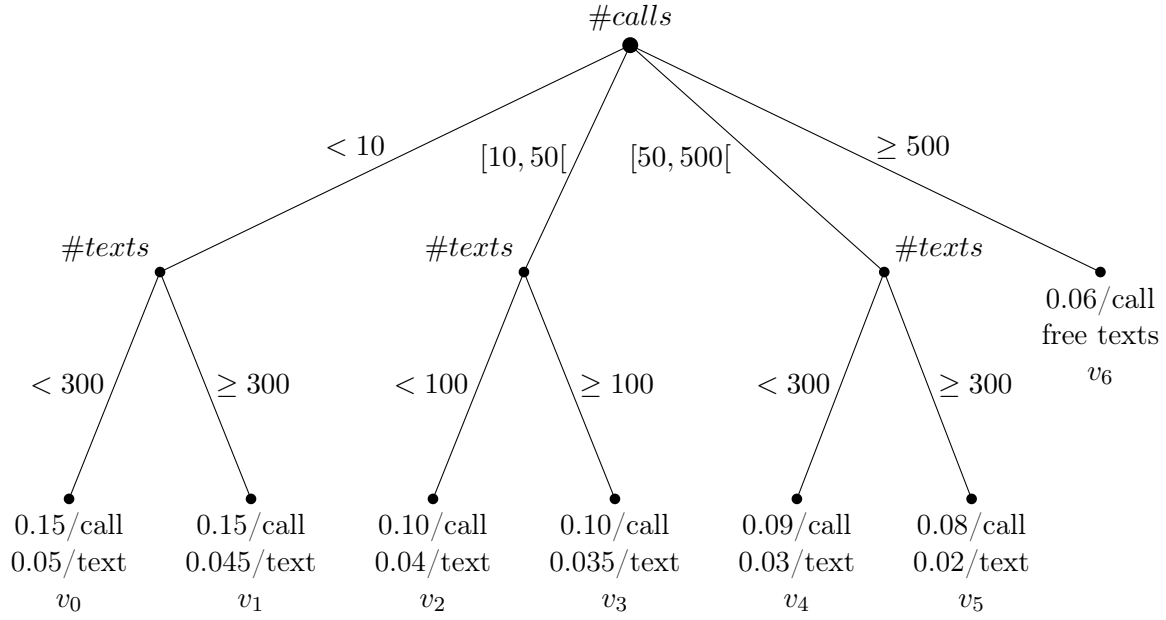
1.2 `computeBill` method

```
float computeBill(Client client) {
```

The responsibility of `computeBill` method is to determine the value to pay for a client taking into account all communications made by the client through all of his registered mobile phones

1.2.1 Test Pattern – Combinational Function

1.2.2 Decision Tree



1.2.3 Domain Matrices

v_0			**Test Cases**			
Variable	**Condition**	**Type**	—	1	—	2
#calls	< 10	ON	10			
		OFF		9		
	Typical	IN			8	7
#texts	< 300	ON			300	
		OFF				299
	Typical	IN	147	204		
Expected Result			v_3	11.55	v_1	16.00

Table 1: v_0 domain matrix

v_1			**Test Cases**			
Variable	**Condition**	**Type**	—	3	4	—
#calls	< 10	ON	10			
		OFF		9		
	Typical	IN			6	5
#texts	≥ 300	ON			300	
		OFF				299
	Typical	IN	320	400		
Expected Result			v_3	19.35	14.40	v_0

Table 2: v_1 domain matrix

v_2			**Test Cases**					
Variable	**Condition**	**Type**	5	—	—	6	—	7
#calls	≥ 10	ON	10					
		OFF		9				
	< 50	ON			50			
		OFF				49		
	Typical	IN					22	35
#texts	< 100	ON					100	
		OFF						99
	Typical	IN	48	20	33	15		
Expected Result			2.92	v_0	v_4	5.50	v_3	7.46

Table 3: v_2 domain matrix

v_3			**Test Cases**					
Variable	**Condition**	**Type**	8	—	—	9	10	—
#calls	≥ 10	ON	10					
		OFF		9				
	< 50	ON			50			
		OFF				49		
	Typical	IN					12	44
#texts	≥ 100	ON					100	
		OFF						99
	Typical	IN	148	220	333	414		
Expected Result			6.18	v_0	v_5	15.49	4.70	v_2

Table 4: v_3 domain matrix

v_4			**Test Cases**					
Variable	**Condition**	**Type**	11	—	—	12	—	13
#calls	≥ 50	ON	50					
		OFF		49				
	< 500	ON			500			
		OFF				499		
	Typical	IN					142	51
#texts	< 300	ON					300	
		OFF						299
	Typical	IN	240	189	98	10		
Expected Result			11.70	v_3	v_6	45.21	v_5	13.56

Table 5: v_4 domain matrix

v_5			**Test Cases**					
Variable	**Condition**	**Type**	14	—	—	15	16	—
#calls	≥ 50	ON	50					
		OFF		49				
	< 500	ON			500			
		OFF				499		
	Typical	IN					200	60
#texts	≥ 300	ON					300	
		OFF						299
	Typical	IN	314	500	616	404		
Expected Result			10.28	v_3	v_6	48.00	22.00	v_4

Table 6: v_5 domain matrix

v_6			**Test Cases**	
Variable	**Condition**	**Type**	17	—
#calls	≥ 500	ON	500	
		OFF		499
Expected Result			30.00	v_4/v_5

Table 7: v_6 domain matrix

2 Class-Scope Tests

2.1 `Client` class

2.2 `Mobile` class